# Diamond Price Prediction

## CONTENT :



**NOTE :** The dataset which has been used in this project is taken from the source named “Kaggle” We can change the dataset for making preditions form another source.

## Project’s Aim :

To predict the price of the diamond.

**Name Of The Project** : Diamond price prediction

## Project Description :

The aim of this project was to predict the price of the diamond based on carat, color, cut and clarity. This model builds high level accuracy . Precious stones like diamond are in high demand in the investment market due to their monetary rewards.

The Random Forest algorithm is used to predict the diamond price and determine which attribute affects the most.

## Introduction :

Diamond is a solid form of carbon element that present in crystal structure that known as diamond cubic making it unique. Diamond is known with their hardness, good thermal conductivity, high index of refraction, high dispersion, and adamantine luster. The high luster gives diamond the ability to reflect lights that strikes on their surface thus giving them the 'sparkle'.

Colour and clarity determine the price of diamond to be selected as jewellery gems. Jewellery diamonds have the lowest number of specific gravity with it happens to be very close to 3.52 with minimal impurities and defects. Quality of diamonds that are made into jewellery gems are determined by color, cut, clarity and carat weight. Diamond attributes are as follows:

• **Colour**: Most quality diamond ranging from colorless to slightly yellow, brown or grey. The highest and most valuable diamonds is the one that are completely colorless.


• **Clarity**: An ideal diamond is free from fracture and particles of foreign material within the gems as low clarity gems tends to degrade the appearance, reduce the strength of the stone thus lower its value.

• **Cut**: Quality of designs and craftsmanship determines the appearance of diamonds that later determines the price. Angles of facets cut, proportions of design and quality of polishing determines face-up appearance, brilliance, scintillation, pattern and fire. A perfect diamond stones are perfectly polished, highly reflective, emit maximum amount of fire, faceted faces equal in size and the edges meet perfectly also identical in shape.


• **Carat**: A unit of weight equal to 1/5 of a gram or 1/142 of an ounce. Small diamonds are usually cost less per carat because of its common presences.

Another category of diamonds that are currently becoming a trend among diamond jewellery lovers are colored diamonds that occur in variety of hues such as red, pink, yellow, orange, purple, blue, green, and brown. The quality of this diamond's type is determined by intensity, purity, and quality of their colour, which, the most saturated and vivid colour hold a greater price.

## Dataset Details :

• **Title**: Diamonds

• **Year**: 2017

• **Source**: Kaggle Website (https://www.kaggle.com/datasets/shivam2503/diamonds?datasetId=1312&sortBy=voteCount&searchQuery=class)

• **Purpose of Dataset**: A great simple dataset for beginners who is learning to work in data analysis and visualization.

• **Content**: Diamond attributes of price, carat, cut, color, clarity, length, width, depth, total depth percentage, width of top of diamonds.

| Attribute | Description |
|---|---|
| price | in US dollars ($326 - $18,823) |
| carat | weight of the diamond (0.2 - 5.01) |
| cut | quality of the cut (Fair, Good, Very Good, Premium, Ideal) |
| color | diamond colour, from J (worst) to D (best) |
| clarity | a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best)) |
| x | length in mm (0 - 10.74) |
| y | width in mm (0 - 58.9) |
| z | depth in mm (0 - 31.8) |
| depth | total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43 - 79) |
| table | width of top of diamond relative to widest point (43 - 95) |

• **Structure**: Mainly consist of integers, floating point values also string.

• **Summary**: This dataset describes attributes of the 54,000 diamonds together with the price so the dataset can be make used to propose suitable linear regression or just normal exploratory data analysis.

## Problem Statement :

Diamond gems is one of the most popular gems in entire world. This valuable gem can be worth from as low as hundreds and up to millions. However, no clear guidelines or understanding on the determination of diamond's price in the market. Therefore, exploring which attributes determine the value of a diamond gems may helps with predicting the price of the diamonds.

# Objectives

To explore which attributes contribute to the price range in diamond gems.

To predict the price of diamond gems from corresponding attributes

To determine the characteristics that affect the cut of the diamond.

## Input :

cut, colour, clarity, carat.

## Output :

Approximate Price of the diamond

**Let's get started to build the model based on the following steps:**

1. Import Required Packages
2. Load the dataset
3. Perform the exploratory data analysis (EDA)
4. Prepare the dataset for training
5. Create a regression model
6. Train the model to fit the data
7. Make predictions using the trained model

## Used libraries:

- **Pandas:**
  pandas is a python library, It is used to manipulate , transform and visualize data easily and efficiently. In my project pandas is used to reading the data and do some manipulations on dataset.

- **Numpy:**

numpy library is the core library for scientific computation in python. It provides a high performance multidimentional array object and tools for working with these arrays. But numpy is a fixed size and it must be of the same data type. Here I am using numpy for some mathematical operations, slicing and also for slicing.

- **Matplotlib:**
  matplotlib is an visualization library on 2D plots of arrays. And it contains several plots like line ,bar ,scatter, histograms, etc. matplotlib comes with a wide variety of plots which helps to understand trends, patterns, and to make correlations. Here I am using matplotlib for plotting the data of my dataset.

- **Seaborn:**
  seaborn supports complex visualizations of data. It is built on matplotlib and works with pandas dataframes . matplotlib is better for basic plots while seaborn is better for more advanced statical plots (ex:distplot similar as histograms but by defaultly, it generates a Gaussian kernel density estimate). Here I am using seaborn for plot a lmplot on dataset.

- **Sklearn:**
  scikit learn is the most useful and robust library. It provides a selection of efficient tools and statistical modeling including classification, regression, clustering. Here I am unsing sklearn for data splitting , training and testing. And also for predict the output for given input from dataset.

## Used Alogirthms

## Linear regression :

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

## K-Nearest Neighbor (KNN Algorithm ):

K-nearest neighbor is one of the simplest machine learning algorithm based on supervised Learning technique.

Knn algorithm assumes the similarity between the new data point based on the similarity.this means when new data appers then it can be easily classified into a well suite category by using knn algorithm.

**Steps:**

1. Select the number k of the neighbors.
2. Calculate the Euclidean distance of k number of neighbors.
3. Take the k nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors,count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.

## DECISION TREE :

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

# RANDOM FOREST ALGORITHM :

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process Of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

## Problem Statement :

Diamond gems is one of the most popular gems in entire world. This valuable gem can be worth from as low as hundreds and up to millions. However, no clear guidelines or understanding on the determination of diamond’s price in the market. Therefore, exploring which attributes determine the value of a diamond gems may helps with predicting the price of the diamonds.

## PROGRAM :

### Data Exploration and Preprocessing :

In [4]:
```
import pandas as pd
import numpy as np
data=pd.read_csv(r"C:\Users\rgukt iiit\Downloads\diamonds.csv")
data
```

Out[4]:

| | Unnamed: 0 | carat | cut | color | clarity | depth | table | price | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0.23 | Ideal | E | SI2 | 61.5 | 55.0 | 326 | 3.95 | 3.98 | 2.43 |
| 1 | 2 | 0.21 | Premium | E | SI1 | 59.8 | 61.0 | 326 | 3.89 | 3.84 | 2.31 |
| 2 | 3 | 0.23 | Good | E | VS1 | 56.9 | 65.0 | 327 | 4.05 | 4.07 | 2.31 |
| 3 | 4 | 0.29 | Premium | I | VS2 | 62.4 | 58.0 | 334 | 4.20 | 4.23 | 2.63 |
| 4 | 5 | 0.31 | Good | J | SI2 | 63.3 | 58.0 | 335 | 4.34 | 4.35 | 2.75 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 53935 | 53936 | 0.72 | Ideal | D | SI1 | 60.8 | 57.0 | 2757 | 5.75 | 5.76 | 3.50 |
| 53936 | 53937 | 0.72 | Good | D | SI1 | 63.1 | 55.0 | 2757 | 5.69 | 5.75 | 3.61 |
| 53937 | 53938 | 0.70 | Very Good | D | SI1 | 62.8 | 60.0 | 2757 | 5.66 | 5.68 | 3.56 |
| 53938 | 53939 | 0.86 | Premium | H | SI2 | 61.0 | 58.0 | 2757 | 6.15 | 6.12 | 3.74 |
| 53939 | 53940 | 0.75 | Ideal | D | SI2 | 62.2 | 55.0 | 2757 | 5.83 | 5.87 | 3.64 |

53940 rows × 11 columns

[5]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import pandas.util.testing as tm
```

In [6]:
```
data.head()
```

Out[6]:

| | Unnamed: 0 | carat | cut | color | clarity | depth | table | price | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0.23 | Ideal | E | SI2 | 61.5 | 55.0 | 326 | 3.95 | 3.98 | 2.43 |
| 1 | 2 | 0.21 | Premium | E | SI1 | 59.8 | 61.0 | 326 | 3.89 | 3.84 | 2.31 |
| 2 | 3 | 0.23 | Good | E | VS1 | 56.9 | 65.0 | 327 | 4.05 | 4.07 | 2.31 |
| 3 | 4 | 0.29 | Premium | I | VS2 | 62.4 | 58.0 | 334 | 4.20 | 4.23 | 2.63 |
| 4 | 5 | 0.31 | Good | J | SI2 | 63.3 | 58.0 | 335 | 4.34 | 4.35 | 2.75 |

In [7]:
```
data.describe()
```

| | Unnamed: 0 | carat | depth | table | price | x | |
|---|---|---|---|---|---|---|---|
| count | 53940.000000 | 53940.000000 | 53940.000000 | 53940.000000 | 53940.000000 | 53940.000000 | 53940. |
| mean | 26970.500000 | 0.797940 | 61.749405 | 57.457184 | 3932.799722 | 5.731157 | 5. |
| std | 15571.281097 | 0.474011 | 1.432621 | 2.234491 | 3989.439738 | 1.121761 | 1. |
| min | 1.000000 | 0.200000 | 43.000000 | 43.000000 | 326.000000 | 0.000000 | 0. |
| 25% | 13485.750000 | 0.400000 | 61.000000 | 56.000000 | 950.000000 | 4.710000 | 4. |
| 50% | 26970.500000 | 0.700000 | 61.800000 | 57.000000 | 2401.000000 | 5.700000 | 5. |
| 75% | 40455.250000 | 1.040000 | 62.500000 | 59.000000 | 5324.250000 | 6.540000 | 6. |
| max | 53940.000000 | 5.010000 | 79.000000 | 95.000000 | 18823.000000 | 10.740000 | 58. |

]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 11 columns):
Unnamed: 0    53940 non-null int64
carat         53940 non-null float64
cut           53940 non-null object
color         53940 non-null object
clarity       53940 non-null object
depth         53940 non-null float64
table         53940 non-null float64
price         53940 non-null int64
x             53940 non-null float64
y             53940 non-null float64
z             53940 non-null float64
dtypes: float64(6), int64(2), object(3)
memory usage: 4.5+ MB
```

```
]: data.isnull().sum()
```

```
Unnamed: 0    0
carat         0
cut           0
color         0
clarity       0
depth         0
table         0
price         0
x             0
y             0
z             0
dtype: int64
```

```
data=data.drop(['depth','table','x','y','z','Unnamed: 0'],axis=1)
```

```
data
```

| | carat | cut | color | clarity | price |
|---|---|---|---|---|---|
| 0 | 0.23 | Ideal | E | SI2 | 326 |
| 1 | 0.21 | Premium | E | SI1 | 326 |
| 2 | 0.23 | Good | E | VS1 | 327 |
| 3 | 0.29 | Premium | I | VS2 | 334 |
| 4 | 0.31 | Good | J | SI2 | 335 |
| ... | ... | ... | ... | ... | ... |
| 53935 | 0.72 | Ideal | D | SI1 | 2757 |
| 53936 | 0.72 | Good | D | SI1 | 2757 |
| 53937 | 0.70 | Very Good | D | SI1 | 2757 |
| 53938 | 0.86 | Premium | H | SI2 | 2757 |
| 53939 | 0.75 | Ideal | D | SI2 | 2757 |


| | | | | | |
|---|---|---|---|---|---|
| 53938 | 0.86 | Premium | H | SI2 | 2757 |
| 53939 | 0.75 | Ideal | D | SI2 | 2757 |

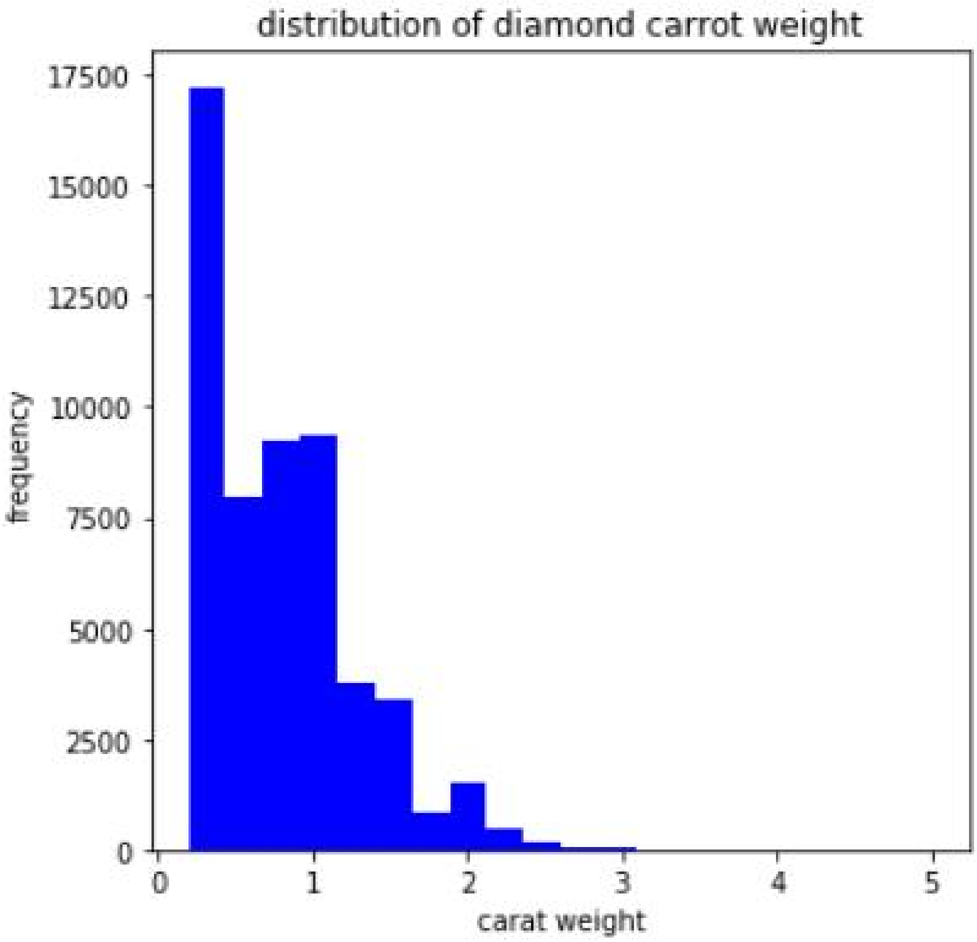

53940 rows × 5 columns

```
data.head()
```

| | carat | cut | color | clarity | price |
|---|---|---|---|---|---|
| 0 | 0.23 | Ideal | E | SI2 | 326 |
| 1 | 0.21 | Premium | E | SI1 | 326 |
| 2 | 0.23 | Good | E | VS1 | 327 |
| 3 | 0.29 | Premium | I | VS2 | 334 |
| 4 | 0.31 | Good | J | SI2 | 335 |

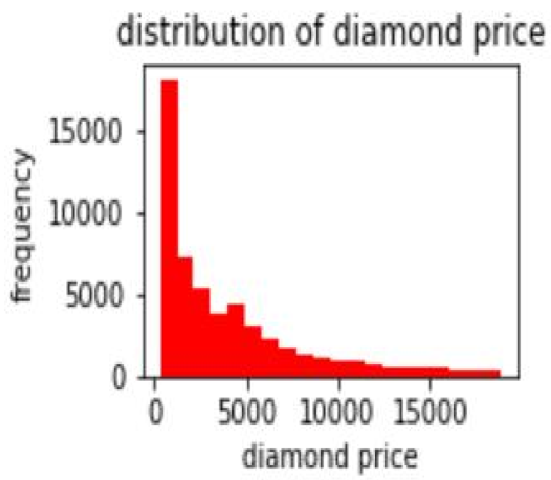
```
plt.figure(figsize=[12,12])
plt.subplot(221)
#carat weight distribution
plt.hist(data['carat'],bins=20,color='b')
plt.xlabel("carat weight")
plt.ylabel("frequency")
plt.title("distribution of diamond carrot weight")
```

```
Text(0.5, 1.0, 'distribution of diamond carrot weight')
```



In [17]:
```python
plt.subplot(221)
#distribution of price values
plt.hist(data['price'],bins=20,color='r')
plt.xlabel("diamond price")
plt.ylabel("frequency")
plt.title("distribution of diamond price")
```

Out[17]: Text(0.5, 1.0, 'distribution of diamond price')




```python
data.head(1)
```

| | carat | cut | color | clarity | price |
|---|---|---|---|---|---|
| 0 | 0.23 | Ideal | E | SI2 | 326.0 |

```python
from sklearn.preprocessing import LabelEncoder
l1=LabelEncoder()
label=l1.fit_transform(data["cut"])
l1.classes_
```

array(['Fair', 'Good', 'Ideal', 'Premium', 'Very Good'], dtype=object)

```python
label
```

array([2, 3, 1, ..., 4, 3, 2])

```python
data["cut_label"]=label
data.head(2)
```

| | carat | cut | color | clarity | price | cut_label |
|---|---|---|---|---|---|---|
| 0 | 0.23 | Ideal | E | SI2 | 326.0 | 2 |
| 1 | 0.21 | Premium | E | SI1 | 326.0 | 3 |

```python
l2=LabelEncoder()
label1=l2.fit_transform(data["clarity"])
data["clarity_label"]=label1
data.head(2)
```

| | carat | cut | color | clarity | price | cut_label | clarity_label |
|---|---|---|---|---|---|---|---|
| 0 | 0.23 | Ideal | E | SI2 | 326.0 | 2 | 3 |
| 1 | 0.21 | Premium | E | SI1 | 326.0 | 3 | 2 |

```python
data["color"]=data["color"].map({'D':1,'E':2,'F':3,'G':4,'H':5,'I':6,"J":7,"NA":8})
data["color"].fillna(0)
```

```
0        2
```

```
0        2
1        2
2        2
3        6
4        7
        ..
53935    1
53936    1
53937    1
53938    5
53939    1
Name: color, Length: 53940, dtype: int64
```

```python
data["color"].isnull().sum()
```

```
0
```

```python
data.head(2)
```

| | carat | cut | color | clarity | price | cut_label | clarity_label |
|---|---|---|---|---|---|---|---|
| 0 | 0.23 | Ideal | 2 | SI2 | 326.0 | 2 | 3 |
| 1 | 0.21 | Premium | 2 | SI1 | 326.0 | 3 | 2 |

```
y=data["price"]
y.head(1)
```

```
0    326.0
Name: price, dtype: float64
```

```
x=data.drop(["price","cut","clarity"],axis=1)
```

```
x.head(1)
```

# Training And Splitting the data

| | carat | color | cut_label | clarity_label |
|---|---|---|---|---|
| 0 | 0.23 | 2 | 2 | 3 |

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=4
```

```
len(x_train)
```

```
43152
```

```
len(y_test)
```

```
10788
```

```
len(data)
```

```
53940
```

```
data.head()
```

| | carat | cut | color | clarity | price | cut_label | clarity_label |
|---|---|---|---|---|---|---|---|
| 0 | 0.23 | Ideal | 2 | SI2 | 326.0 | 2 | 3 |
| 1 | 0.21 | Premium | 2 | SI1 | 326.0 | 3 | 2 |
| 2 | 0.23 | Good | 2 | VS1 | 327.0 | 1 | 4 |
| 3 | 0.29 | Premium | 6 | VS2 | 334.0 | 3 | 5 |
| 4 | 0.31 | Good | 7 | SI2 | 335.0 | 1 | 3 |

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
pred=reg.predict(x_test)
```

```
from sklearn.metrics import r2_score
lr=r2_score(y_test,pred)*100
print(lr)
```

```
87.76517206528275
```

```
from sklearn.tree import DecisionTreeRegressor
reg=DecisionTreeRegressor()
reg.fit(x_train,y_train)
pred1=reg.predict(x_test)
```

```
dtr=r2_score(y_test,pred1)*100
print(dtr)
```

```
97.13740996264994
```

```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor(n_estimators=50)
rf.fit(x_train,y_train)
pred2=rf.predict(x_test)
```

```
rfr=r2_score(y_test,pred2)*100
print(rfr)
```

```
97.75012499581325
```

```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor(n_neighbors=5)
knn.fit(x_train,y_train)
pred3=knn.predict(x_test)
```

```
knn=r2_score(y_test,pred3)*100
print(knn)
```

```
97.50021264213912
```

```python
print('LinearRegression',lr)
print('Decision Tree',dtr)
print('Random Forest',rfr)
print('KNeighbors',knn)
```

```
LinearRegression 87.76517206528275
Decision Tree 97.13740996264994
Random Forest 97.75012499581325
KNeighbors 97.50021264213912
```

```python
def prediction():
    carat=(input("enter the carat values:"))
    color=int(input("enter the color value:"))
    clarity=int(input("enter the clarity value:"))
    cut=int(input("enter the cut value:"))
    for i in [carat,color,clarity,cut]:
        if(i==0):
            print("no price")
            break
        elif(int(i)<0):
            print("undefined")
            break
    else:
            price=rf.predict([[carat,color,clarity,cut]])
            print("$",price)
prediction()
```

```
enter the carat values:1
enter the color value:2
enter the clarity value:3
enter the cut value:4
$ [6121.97]
```

```
enter the values:0
enter the value:0
enter the value:0
enter the value:0
no price
```

```
enter the values:-1
enter the value:-2
enter the value:-3
enter the value:-4
undefined
```

```python
def prediction():
    carat=(input("enter the carat values:"))
    color=int(input("enter the color value:"))
    clarity=int(input("enter the claruty value:"))
    cut=int(input("enter the cut value:"))
    for i in [carat,color,clarity,cut]:
        if(i==0):
            print("no price")
            break
    else:
            price=rf.predict([[carat,color,clarity,cut]])
            print("$",price)
prediction()
```

```
enter the carat values:0.2
enter the color value:3
enter the claruty value:4
enter the cut value:2
$ [3223.44]
```